# Spinner Widget

The spinner widget can be used as an activity indicator, showing users that your app is busy performing a task that will take some time to complete.

When the spinner widget is visible, it spins around. You can use the visible property of the widget to control when it's displayed to the user. Normally, it would make sense to display it in front of other content displayed on the screen.

# Creating a Spinner

A spinner widget can be created by dragging it out from the Tools Palette, where it appears with the following icon:

Alternatively it can be created in script using:

```
create widget as "com.livecode.widget.spinner"
```

# Using the Spinner

Here are some tips that may help you use the spinner effectively.

## Ensuring the Spinner Actually Spins

The spinner widget will only update (rotate to its next notch) when the engine gets to do event processing, such as during a wait with messages statement. There are several commands and functions which implicitly wait with messages internally, such as the url chunk.

However, if you are doing a slow computation, such as parsing a text file or loading data from a database, you may wish to add `wait 0 with messages` statement each time you make progress, so that the spinner continues to spin, and your users don't think your app has frozen.

## Making it Clear that the App is "Busy"

You may wish to hide or "dim" your app's user interface while the app is busy. For example, your app could use a "splash screen" card that displays a logo and a spinner widget while loading its resources and preparing to run. This helps by providing additional visual cues to the user that the app's user interface is not ready to respond yet.

## Cancelling a Long-Running Activity

Sometimes, you may have a very slow process that you can safely abandon. It can be useful to provide a “Cancel” button. Suppose you have a stack with a cancelButton and a spinnerWidget. You can create a cancellable process using a script-local variable to store whether the process should continue:

```
-- In the script of the cancelButton
on mouseUp
   send "cancelOperation"
end mouseUp

-- In the stack script
local sCancelled

command doOperation
   set the visible of control "spinnerWidget" to true
   set the visible of control "cancelButton" to true
   put false into sCancelled

   repeat for each item to process
      -- (do something with the item)

      -- allow the spinner widget to spin, and
      -- for cancel button clicks to be handled
      wait 0 with messages
      if sCancelled then
         exit repeat
      end if
   end repeat

   if not sCancelled then
      -- (update the UI to reflect the results)
   end if
   set the visible of control "spinnerWidget" to false
   set the visible of control "cancelButton" to false
end doOperation
```

# Indicating Background Processing

Sometimes your app may need to do some background processing, such as synchronising its state with a server, without needing to block the user interface. You may wish to use a spinner widget in a peripheral part of the user interface, such as a status bar, to provide an unobtrusive indication that this is going on.

For example, an e-mail app might regularly check for new messages, but that shouldn’t necessarily block the user from carrying on reading or writing e-mails.